NPS-MA-01-001

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Support of JCATS Limited V&V

by

James G. Taylor
Beny Neta

September 2001

Approved for public release; distribution is unlimited.

Prepared for: Dismounted Battlespace Battle Laboratory
Ft Benning, GA

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM David R. Ellison
Superintendent

R. Elster
Provost

This report was prepared for Naval Postgraduate School and funded by Dismounted Battlespace Battle Laboratory.

This report was prepared by:

James G. Taylor
Professor
Operations Research Department

Beny Neta
Professor
Mathematics Department

Reviewed by:

Michael A. Morgan, Chair
Mathematics Department

Released by:

D. W. Netzer
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE

Form approved

OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2001	1 October 2000 - 30 September 2001

4. TITLE AND SUBTITLE

Support of JCATS Limited V&V

5. FUNDING

MIPROCNPSJV033

6. AUTHOR(S)

James G. Taylor and Beny Neta

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-MA-01-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Dismounted Battlespace Battle Laboratory
Simulation Center
Bldg. 2868A Way Street
Ft. Benning, GA 31905

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Approved for public release; distribution is unlimited.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words.)

The goal of this study effort was to assess the ability of the Joint Conflict and Tactical Simulation (JCATS) to simulate the capabilities of non-lethal weapons (NLW) and to provide a product that can be incorporated into the full VV&A of JCATS. This work investigated the first 32 algorithms on the JNLWD V&V Priority List. It evaluated JCATS algorithms in two ways:

(1) verification of computer code against algorithm documentation,

(2) appropriateness of algorithms within context of U.S. Army current model standards.

All 32 algorithms were verified, with very few discrepancies with the documentation being found. Of these 32 algorithms, only 25 were documented already by LLNL in the JCATS Algorithm Manual so documentation for the remaining 7 was developed with the help of LLNL from documentation internal to the JCATS computer code. Evaluation of these algorithms (actually a subset of five or so key algorithms) within the context of a compendium of algorithms developed for the Close Combat Tactical Trainer (CCTT) developed by AMSAA revealed that several key algorithms (particularly target acquisition) should be upgraded, if possible. This research also revealed a document that could be used to provide the theoretical basis of most of the AMSAA algorithms, particularly those for attrition. Such a document was never available to LLNL. Although some key algorithms should be upgraded (mainly because of modeling and simulation developments of the last five years or so), all JCATS algorithms (including its target-acquisition algorithm) were at one time more than adequate for analysis purposes. Moreover, overall the algorithms reviewed are deemed to be adequate (particularly in comparison with Janus Army) for playing close combat with non-lethal weapons in urban terrain for purposes of analysis. Further work (particularly along the lines of the issues raised by this work) is necessary, however, to document these modeling issues. Some research is required to better articulate the technical issues raised here, particularly if future V&V efforts are to build on the work at hand.

14. SUBJECT TERMS	15. NUMBER OF PAGES	16. PRICE CODE
limited V&V, algorithms evaluation, target acquisition	46	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

Support of JCATS Limited V&V

by

James G. Taylor
MOVES Academic Group
Naval Postgraduate School
Monterey, CA 93943

and

Beny Neta
Mathematics Department
Naval Postgraduate School
Monterey, CA 93943

ABSTRACT:

The goal of this study effort was to assess the ability of the Joint Conflict and Tactical Simulation (JCATS) to simulate the capabilities of non-lethal weapons (NLW) and to provide a product that can be incorporated into the full VV&A of JCATS. This work investigated the first 32 algorithms on the JNLWD V&V Priority List. It evaluated JCATS algorithms in two ways:

(1) verification of computer code against algorithm documentation,

(2) appropriateness of algorithms within context of U.S. Army current model standards.

All 32 algorithms were verified, with very few discrepancies with the documentation being found. Of these 32 algorithms, only 25 were documented already by LLNL in the JCATS Algorithm Manual so documentation for the remaining 7 was developed with the help of LLNL from documentation internal to the JCATS computer code. Evaluation of these algorithms (actually a subset of five or so key algorithms) within the context of a compendium of algorithms developed for the Close Combat Tactical Trainer (CCTT) developed by AMSAA revealed that several key algorithms (particularly target acquisition) should be upgraded, if possible. This research also revealed a document that could be used to provide the theoretical basis of most of the AMSAA algorithms, particularly those for attrition. Such a document was never available to LLNL. Although some key algorithms should be upgraded (mainly because of modeling and simulation developments of the last five years or so), all JCATS algorithms (including its target-acquisition algorithm) were at one time more than adequate for analysis purposes. Moreover, overall the algorithms reviewed are deemed to be adequate (particularly in comparison with Janus Army) for playing close combat with non-lethal weapons in urban terrain for purposes of analysis. Further work (particularly along the lines of the issues raised by this work) is necessary, however, to document these modeling issues. Some research is required to better articulate the technical issues raised here, particularly if future V&V efforts are to build on the work at hand.

Introduction:

The Dismounted BattleSpace Battlelab (DBBL) is planning to conduct a limited verification and validation (V&V) study of the non-lethal capabilities of the Joint Conflict and Tactical Simulation (JCATS) model. The goal of this study effort is (1) to assess the model's ability to simulate the capabilities of non-lethal weapons (NLW) and (2) to provide a product that can be incorporated into the full VV&A of JCATS.

Statement of Work:

Assist in the conduct of a limited V&V of the non-lethal capabilities of JCATS.

1. The first 32 algorithms on the proposed JNLWD V&V Priority List of Algorithms (see Appendix 1) will be reviewed in detail and verified that they are appropriately implemented in the JCATS computer code.
2. Attention (but at a lower level of priority) is also to be paid to algorithms validity (particularly the algorithms for simulating the capabilities of NLW) and whether the JCATS algorithms satisfy Army model standards in a fashion consistent with JNLWD intended use.

Algorithm Evaluation:

This work investigated the first 32 algorithms on the JNLWD V&V Priority List. It evaluated JCATS algorithms in the following two ways:

(1) verification of computer code against algorithm documentation,
(2) appropriateness of algorithm.

The first aspect (i.e. verification of algorithm implementation in JCATS computer code) is straightforward and a well-accepted part of the V&V (Verification and Validation) process. It does not need further discussion. Results of this algorithm verification are given below.

The investigation of the appropriateness of JCATS algorithms was a more subtle task, and only a few key algorithms were investigated. Some key algorithms investigated (e.g. target acquisition, assessment of direct-fire-engagement outcomes) were found to be in need of upgrade. To be sure, those algorithms that were chosen for investigation were those that were suspected of needing such upgrade. In all fairness, though, the same could be said about Janus Army (e.g. its use of independent rounds for direct-fire-engagement-outcome assessment). Thus, some upgrading (particularly for target acquisition) should be done, but it is the opinion of these authors that JCATS is quite comparable to other current high-resolution Monte-Carlo combat simulations that are currently used for analytical work in DoD. Although a legacy model (for which there is no funding for further model development unless specifically paid for by a user), JCATS may well be as good as other leading simulations that could be used to investigate close combat with non-lethal munitions, especially for military operations in urban terrain.

The need to investigate the appropriateness of the JCATS algorithms in the first place came from consideration of the target-acquisition algorithms. It was found that the algorithm for the optical-sensor model was an obsolete one that the Army had replaced since the original development of Janus (from which JCATS has descended). In fact, since the original development of Janus the

Army had developed an entire program of model standards[1] had been developed, and the Army had apparently not kept Lawrence Livermore National Laboratory (LLNL) explicitly informed of these developments and others[2]. Consequently, it was found that JCATS was using a number of algorithms at variance with current Army model standards. It is recommended that major JCATS algorithms (see below) be brought into conformity with Army model standards (at least where it appears to make a significant difference in results).

Verification of Computer Code Against Algorithm Documentation:

Our verification work was to compare the JCATS Algorithm Manual (draft version 2.0.0) written by the Conflict Simulation Laboratory of LLNL (report number UCRL-MA-135117 DR, dated 30 September 1999) to the code. Beny Neta visited Lawrence Livermore on 6 April 2000 and met with the principals at the Lab. He was given full access to the code and help from Hal Brand to answer any questions that he had. At the end of the day he was given a hard copy of the following algorithms: NVEOL Thermal Model, Enhanced Lighting and FASCAM effects (which are not in the algorithm manual).

The code for specific algorithms was received by mail (hard copy only) upon request. We have checked that the code agrees with the algorithm manual. We have not run the code, since only hard copy was released to us. Clearly we have made the comparison only for those algorithms for which we had a description in the algorithm manual (see Appendix 1). All but three algorithms agree.

For algorithm 4, NVEOL Optical Sensor Scan we have found several typos and we are including the modified algorithm as appendix 2. For algorithm 8, Assess Hit Internal, we found a typographical error in the manual. The code checks if moFPk ≥ 0, but the manual (page 3-2, lines 16 and 20) by mistake had if moFPk ≤ 0.

For algorithm 19, Engage by Direct Fire, we found a discrepancy in computing median_rounds. The code takes the integer part of (SSPK*100+0.5), i.e. rounds the number and the algorithm manual takes the integer part of (SSPK*100), i.e. chops the number. I have talked to Hal Brand about these two and he said that the algorithm manual would be modified to agree with the code. In our opinion this is the appropriate remedy.

The BEAM weapon algorithm was verified, after we received a write-up from LLNL. We include this write-up as Appendix 6. We visited LLNL again on 24-25 August to complete the V&V for those 5 algorithms not written in the manual. We managed to get three algorithms out of the 5 done. Algorithm 5, NVEOL Thermal Sensor Scan is now given as appendix 3. Algorithm 20, Planned Direct Fire and algorithm 21, Planned Indirect Fire are given in Appendices 5 and 4, respectively.

During another visit on 25-29 September we completed the V&V of the ground movement algorithms (numbered 22-31) and the other undocumented algorithms (5, 17). We made some changes to algorithm #24 (Trafficability Factor). The bullets concerning Fence and Building Components should not be there. In algorithm #25 (Calculation of slope) we modified the formula for speed factor (SF) to read as follows:

[1] As a consequence of the creation of the Defense Modeling and Simulation Office (DMSO) in 1991 and subsequent formation of the Army Model and Simulation Office (AMSO) in 1992. There are currently 19 different model standards categories.

[2] For example, the development of various compendia of algorithms for use in Army models and simulations.

$$SF=(\ln(100*|slope|)-\ln(MaxSlope))/(\ln(100*|slope|)*(1-\ln(MaxSlope))).$$

Algorithm #28 (Fatigue factor) is given in Appendix 9. Algorithm #5 (Enhanced Lighting) is now given as Appendix 7 and algorithm #17 (FASCAM) is in Appendix 8.

Summary of Findings on Algorithm Verification:

- Total of 32 algorithms
  24 documented and 8 are undocumented
  We have received documentation for 1 and generated 7 more with the help of LLNL
  We have verified all 32 algorithms.

The results of the verification are as follows:

- Algorithm 1, Line of sight - done
- Algorithm 2, general sensor scan - done
- Algorithm 3, general sensor sweep - done
- Algorithm 4, NVEOL Optical Sensor Scan – corrected some typos in the algorithm manual
- Algorithm 5, NVEOL Thermal sensor scan – algorithm written with the help of Hal Brand (LLNL)
- Algorithm 6, Enhanced lighting – written with help of LLNL
- Algorithm 7, assess shot - done
- Algorithm 8, Assess hit internal - we found a typographical error in the manual. The code checks if moFPk ≥ 0, but the manual (page 3-2, lines 16 and 20) by mistake had if moFPk ≤ 0.
- Algorithm 9, do secondary suppression - done
- Algorithm 10, assess secondary suppression - done
- Algorithm 11, detonate - done
- Algorithm 12, assess impact - done
- Algorithm 13, handle suppression -done
- Algorithm 14, is suppressed - done
- Algorithm 15, HE effect - done
- Algorithm 16, ICM effect -done
- Algorithm 17, FASCAM – written with help of LLNL
- Algorithm 18, target by direct fire - done
- Algorithm 19, Engage by Direct Fire, we found a discrepancy in computing median_rounds. The code takes the integer part of (SSPK*100+0.5), i.e. rounds the number and the algorithm manual takes the integer part of (SSPK*100), i.e. chops the number. I have talked to Hal Brand about these two and he said that the algorithm manual would be modified to agree with the code.
- Algorithm 20, Planned Direct Fire – written with the help of LLNL
- Algorithm 21, Planned Indirect Fire – written with the help of LLNL
- Algorithm 22, length of hop - done
- Algorithm 23, calculation of speed - done
- Algorithm 24, trafficability factor - done
- Algorithm 25, calculation of slope - done
- Algorithm 26, weather factor - done
- Algorithm 27, lighting factor - done
- Algorithm 28, fatigue – written with help of LLNL
- Algorithm 29, encountering a linear object - done

• Algorithm 30, encountering a minefield - done
• Algorithm 31, encountering other objects - done
• Algorithm 32, Beam weapon - we received a write-up from LLNL.

Appropriateness of Algorithms:

The working hypothesis for the evaluation of the appropriateness of current JCATS algorithms was the following: **the algorithms in "The Compendium of Close Combat Tactical Trainer Algorithms..."** (AMSAA Special Publication No. 74, June 1996)[3] (AMSAA [1996a]) **should be the point of departure for the development of JCATS algorithms.** Discussions with key personnel at AMSAA reinforced that this was an appropriate course of action (Carouthers [2000], Dinsmore [2000]). Moreover, this research revealed that the U.S. Army's "Engineering Design Handbook: Army Weapon System Analysis, Part One" (DARCOM [1977]) provides the theoretical justification for many of the algorithms (particularly, the attrition ones) in "The Compendium of CCTT Algorithms."

Algorithms in Need of Upgrade:

The following algorithms should be upgraded (given in order of decreasing priority):
(1) target acquisition (both optical and thermal sensors),
(2) direct-fire attrition,
(3) indirect-fire attrition,
(4) non-lethal weapons (where appropriate).
These algorithms need upgrade because of changes in Army model standards that have occurred since the development of Janus (and subsequently JCATS) (see Appendix 10).

Concerning target acquisition the two-dimensional ACQUIRE methodology (AMSAA [1996a, Section 2], [2000, Section 2]) should be implemented in JCATS. The so-called Night Vision Laboratory (NVL) methodology used by JCATS was replaced by the ACQUIRE methodology in 1993. The ACQUIRE methodology is in Janus (Army) and all other current Army detailed simulations. It should be easy to implement because it utilizes the same equations (with one minor exception) as the NVL methodology but requires modified input data. ACQUIRE had to be developed because of a new generation of Army sensors.

Furthermore, initialization of sensor-target pairs (see Parish and Kellner [1992]) is another feature that must be implemented in ACQUIRE (Dixon [2000], Parish [2000]). This point is not covered in the AMSAA documentation of ACQUIRE, but was repeatedly stressed by key personnel at TRAC-WSMR. It apparently has a significant impact on simulation outcomes (Dixon [2000], Parish [2000]). These two changes in target acquisition are rated as top-priority items to be implemented in JCATS.

Additionally, for many direct-fire weapons (e.g. tanks), including those used in dismounted infantry combat (Carouthers [2000]), a better model for fire assessment (and one that makes a significant difference in combat outcomes (Dinsmore [2000])) is the miss-distance-distribution method (see Appendix 8). AMSAA apparently has data (Carouthers [2000], Dinsmore [2000]) that allows one to play a "variable bias" (see AMSAA [1996a, p. 4-3]) that leads to significantly different

[3] Updated by AMSAA Special Publication No. 97, May 2000 (AMSAA [2000]).

outcomes in many cases than the assumption of independent rounds (see Appendix 11 and also Appendix 12). Appendix 13 discusses on theoretical grounds why the independent-round model is not a good model for many (if not most) cases of practical interest. Also, AMSAA has similar refined methodology to handle cases of burst-fire systems and burst on target (see AMSAA [1996a, Section 4]). Thus, it appears that the adequacy of the assessment algorithms for direct-fire combat in JCATS need further investigation.

There is also concern about the model for impacts points for indirect-fire weapons. The current algorithm in JCATS (Algorithm 21, Planned Indirect Fire) does not appear to be in conformity with the indirect-fire model in the "CCTT Compendium" (AMSAA [1996a, Section 6 (especially Figure 6-4)]), but there was not sufficient time to investigate this important point in any depth.

Also, there is concern about the playing of non-lethal direct-fire weapons with independent rounds. If AMSAA or some other source has data that allows non-lethal weapons to be played along the lines the recommended playing of conventional direct-fire weapons discussed above (see AMSAA [1996a, Section 4]), then this should be done. If sufficient data does not exist at this time (we did not have time to investigate this important point), then independent rounds would appear to be an adequate model.

Summary of Findings on Evaluation of Algorithms:

Although some key algorithms should be upgraded (mainly because of modeling and simulation developments of the last five years or so), all JCATS algorithms (including its target-acquisition algorithm) were at one time more than adequate for analysis purposes. That is the problem with being a state-of-the-art legacy model for which there has been no funding for further development for a number of years. Now would be a good time to make such capital investment. Moreover, all 32 algorithms investigated were essentially verified to agree with documentation (either internal to the computer code[4] or external in the JCATS Algorithm Manual). The authors were quite impressed by this fact.

Overall, the algorithms reviewed in JCATS appear to be of comparable quality as those in other contemporary, comparable high-resolution Monte-Carlo combat simulations (e.g. Janus Army) and therefore adequate for analysis of issues concerning, for example, close combat with non-lethal munitions. However, this fact should not inhibit further research on such combat models, particularly concerning issues encountered in this work (e.g. adequacy of independent-round model to apply to all direct-fire weapons). In fact, further theoretical research is required just to more adequately articulate what the problems are.

REFERENCES:

Office of the Deputy Under Secretary of the Army (Operations Research) (ODUSOR) and Army Model and Simulation Office (AMSO), "Army Model and Simulation Standards Report, FY98," October 1997 (Copy maintained on AMSO website; current address for AMSO Homepage is http://www.amso.army.mil .)

[4] For these algorithms, documentation was developed with the help of LLNL and appears in the appendices to this report.

Mike Carouthers, Army Materiel Systems Analysis Activity (AMSAA), Personal Communication, August 2000.

Alan Dinsmore, Army Materiel Systems Analysis Activity (AMSAA) (Chairman of AMSO Attrition Standards Coordinating Committee (SCC)), Personal Communication, August 2000.

Dave Dixon, TRAC-White Sands Missile Range (TRAC-WSMR) (Chairman of AMSO Acquire Standards Coordinating Committee (SCC)), Personal Communication, August 2000.

Randall M. Parish, TRAC-White Sands Missile Range (TRAC-WSMR), Personal Communication, August 2000.

Randall M. Parish and A.D. Kellner, "Target Acquisition in Janus Army," U.S. Army TRADOC Analysis Command White Sands Missile Range (TRAC-WSMR), White Sands Missile Range, NM, October 1992.

J.G. Taylor, "Flaw in Janus Direct-Fire Assessments," unpublished working paper created on 4/24/99, Naval Postgraduate School, Monterey, CA, 1999. (a)

J.G. Taylor, "Flaw in Janus Direct-Fire Assessments-2" unpublished working paper created on 4/28/99, Naval Postgraduate School, Monterey, CA, 1999. (b)

U.S. Army Materiel Development and Readiness Command (DARCOM), "Engineering Design Handbook: Army Weapon System Analysis, Part One," DARCOM-P-706-101, Alexandria, VA, November 1977.

U.S. Army Materiel Systems Analysis Activity (AMSAA), "The Compendium of Close Combat Tactical Trainer Algorithms, Data, Data Structures, and Generic System Mappings," Special Publication No. 74, Aberdeen Proving Ground, MD, June 1996. (a)

U.S. Army Materiel Systems Analysis Activity (AMSAA), "Compendium of High Resolution Attrition Algorithms," Special Publication No. 77, Aberdeen Proving Ground, MD, October 1996. (b)

U.S. Army Materiel Systems Analysis Activity (AMSAA), "The Compendium of Close Combat Tactical Trainer Algorithms, Data, Data Structures, and Generic System Mappings," Special Publication No. 97, Aberdeen Proving Ground, MD, May 2000.

M. Uzelac, Lawrence Livermore National Laboratory (LLNL), Personal Communication, June 2000.

Appendix 1

In this section we have the prioritized list of algorithms from DBBL. The priority is given in column 2. Algorithms for which there is no write-up in the JCATS Algorithm Manual (draft dated 30 September 1999, version 2.0.0, report number UCRL-MA-135117 DR) are denoted by TBW in column 3. Column 4 indicate if the code verified. Any findings are given in the last column. In column 5 we indicated by x those algorithms we have in hand (hard copy only). Note that we have verified the acquisition algorithm at the Lawrence Livermore National Laboratory (LLNL). The date we requested the algorithm from LLNL is in column 6.

Algorithms	**Proposed JNLWD V&V Prioritized List**	**To Be Written**	Code verified	In hand	requested	
BEAM	**1**	**TBW**				
Acquisition	**1**		v			
Line of Sight	1		v			
General Sensor Scan	2		v			
General Sensor Sweep	3		v			
NVOEL Optical Sensor Scan	4		v			
NVOEL Thermal Sensor Scan	5	TBW		x	1-May	
Enhanced Lighting	6	TBW		x	1-May	
Weapons Effects	**7**		v	x	1-May	
Point Effect Munitions	**7**		v	x	1-May	
assessShot	7		v	x	1-May	
assessHitInternal	8		v	x	1-May	Found typo in manual
doSecondarySuppression	9		v	x	1-May	
assessSecondarySuppression	10		v	x	1-May	
Area Effect Munitions	**11**		v	x	1-May	
detonate	11		v	x	1-May	
assessImpact	12		v	x	1-May	
handleSuppression	13		v	x	1-May	
isSuppressed(mult)	14		v	x	1-May	
HEeffect	15		v	x	1-May	
ICMeffect	16		v	x	1-May	
FASCAMeffect	17	TBW		x	1-May	
BEAM	17	TBW				
Automated Targeting	**18**		v	x	9-Jun	
Target by Direct Fire	18		v	x	9-Jun	
Engage by Direct Fire	19		v	x	9-Jun	Found disagreement with code
Manual Targeting	**20**			x	9-Jun	
Planned Direct Fire	20	TBW		x	9-Jun	
Planned Indirect Fire	21	TBW		x	9-Jun	
Movement	**22**				28-Jun	
Ground	**22**				28-Jun	
Length of Hop	22				28-Jun	
Calculation of speed	23				28-Jun	
Trafficability Factor	24				28-Jun	
Calculation of Slope	25				28-Jun	
Weather Factor	26				28-Jun	
Lighting Factor	27				28-Jun	

Passive Radar
Passive Sonar
Horizon Check

Appendix 2

General Sensor Scan

All entities within sensor range are considered. The following series of tests is applied to each entity that may be acquired.
If the viewer is not a human with peripheral vision enabled and the entity to be acquired is not in the FOR, ignore it.
If the entity to be acquired is active in an aggregate, ignore it.
If the entity to be acquired is mounted, ignore it.
If the sensor is not sonar and the entity to be acquired is under water, ignore it.
If the entity to be acquired is closer than min sensor range or farther than max sensor range, ignore it.
If the entity to be acquired is dead, ignore it. (Show Dead is a special function handled at the client level.)
If fratricide is on and the entity to be acquired is in the viewer's coordination level, ignore it.
If fratricide is off and the entity to be acquired is a friend, ignore it.
If this sensor can only detect moving targets and the entity to be acquired is not moving (its speed is below the moving speed threshold of 0.25m/s), ignore it.
If this sensor is limited as to air, land or marine targets, test to see if the entity to be acquired is in the right area. If not, ignore it.
If the entity to be acquired hasn't been ignored, try to acquire it.
At this point the algorithms diverge depending on the type of sensor. The rest of the algorithm is provided in the following sensor-specific sections.

General Sensor Sweep

A sweep performs the same process as a scan for the existing acquisition list, except that LOS is not checked (it is assumed to be OK). Each entity is re-sensed, and its acquisition level is adjusted up or down. No entities are added to or removed from the acquisition list during a sweep.

NVEOL Optical Sensors

NVEOL optical sensors model the naked eye, binoculars, etc. They perceive in the visible spectrum (.4 - .7μ). The algorithms in JCATS were derived from the Night Vision Electro-Optical Lab (NVEOL) model. **How do we differ?
At the start of the simulation a 128X128 matrix is generated from the NVEOL Detection Map used in JANUS(A) 5.0. The Detection Map consists of one hundred values representing a log normal distribution. JCATS randomly selects from the Detection Map while filling a 128X128 matrix. All viewer/entity pairs in the simulation are then hash mapped to the matrix. This means that for a given simulation run, a given viewer/entity pair will always have the same acquisition threshold value. However, due to the random fill of the matrix, the same viewer/entity pair may (and probably will) have a different threshold in subsequent runs.
Some terms that will be used in the following discussion are:

- threshhold[viewer][entity] is one of a hundred numbers representing a log normal distribution. It is applied to the cycles constants described below for the various levels of detection.
- cyclesN50Detection, cyclesN50Classification, etc., are the bars needed for a 50% probability of the corresponding level of acquisition given infinite time. They are:
- cyclesN50Detection ≡ 1.0
- cyclesN50Classification ≡ 2.0
- cyclesN50Recognition ≡ 3 .5
- cyclesN50Identification ≡ 6.4

NVEOL Optical Sensor Scan

If the tests described in the General Sensor Scan section have been passed, proceed as follows:
If the viewer is under water, no acquisition by NVEOL sensor is possible. Exit.
Check LOS. If blocked, ignore the entity.
If entity to be acquired is within two meters of the sensor, consider it within the FOR.
If the entity to be acquired is not in the FOR and peripheral acquisition is off, ignore it.
If enhanced lighting is on,
    get ln(contrast_at_target) from the Environment and Light models.
Else,
get ln(contrast_at_target) from the weather model. This value is in bars/milliradian. (DATA)
    If the entity is in defilade,
    ln(contrast_at_target) ← ln(contrast_at_target) - 1.0.
optical_size ← sqrt(optical_dimension * height(posture, defilade) * LOS_exposure_fraction) * transmission_factor

- optical_dimension comes from the PhysicalPropertyModel (DATA), and is different for humans versus all other entities.
- height is defined for non-human systems in Scenario Editor/Systems/Vehicle Data tab. For humans, height is 1.75 meters. In both cases it is adjusted for the entity's posture and defilade state.
- LOS_exposure_fraction is the fraction of total height to which the sensor has unobstructed LOS.
- transmission_factor is calculated using PLOSB through intermediate terrain features and smoke, if any.
- PLOSB is the probability that LOS is blocked per 10 meters of this terrain feature and is defined for a given type of terrain in the Terrain Editor.

If range <= 10 meters,
    optical_size ← 100 * optical_size

- range is the distance from the sensor to the entity to be acquired in meters. It is calculated in three dimensions.

ln(contrast_at_sensor) ← ln(contrast_at_target) + ln(extinction(range))

- extinction is loss of contrast resulting from normal atmospheric effects. This value comes from the weather type entered in Scenario Editor/Tools/Scenario Parameters/Environment/Weather Conditions tab and is a function of range.

If (ln(contrast_at_sensor) < ln(minContrast))
    sensitivity(ln(contrast_at_sensor)) ← 0.0.
Else if (ln(contrast_at_sensor) > ln(maxContrast)),
    sensitivity(ln(contrast_at_sensor)) ← maxCyclesPerMilliRadian.
Else,
    sensitivity(ln(contrast_at_sensor)) ← value from slope, intercept calculation.
true_bars ← sensitivity(ln(contrast_at_sensor)) * (optical_size /range) * 1000

- true_bars are the bars of resolution used to determine acquisition level.

If currentSimTime() < weaponsEffectEnd,
    WeaponsEnhancementMultiplier ← weaponsEffectMultiplier.
Else,
    WeaponsEnhancementMultiplier ← 1.0.
If speed > movingTargetSpeed,
    detFactor ← movingTargetSize.
Else,
    detFactor ← 1.0.
detection_bars ← true_bars * weaponEnhancementMultiplier * detFactor

- detection_bars are the bars of resolution used to test for detection.
- weaponEnhancementMultiplier accounts for the increased probability of detecting a system that just fired its weapon.
- detFactor increases the effective size of a moving system.

If the viewer just blinked (is suppressed),
    acquisitionFactor ← acquisitionFactor * reacquisitionFactor

- reacquisitionFactor is defined in Scenario Editor/Tools/Scenario Parameters/Human Factors/Acquisition tab.

If the viewer is moving (speed ≥ 0.25m/s),
    acquisitionFactor ← acquisitionFactor * movingSensorSize

- movingSensorSize is defined in Scenario Editor/Tools/Scenario Parameters/Human Factors/Acquisition tab.

If(detection_bars < threshhold[viewer][entity] * cyclesN50Det), ignore it.

- threshhold[][] is the value from the 128X128 matrix described earlier.

Else,

    acquire it.

If this is a new acquisition (not on the old acquisition list),

    acquisition_priority ← 0.5 * detection_bars for entities outside the FOR, or
    acquisition_priority ← 1.0 * detection_bars for entities inside the FOR.

If the entity is in the FOR,

    if it recently fired its weapon,
    prob_in_FOV ← 1.0

- Just Fired Time is defined in Scenario Editor/Tools/Scenario Parameters/Human Factors/Acquisition tab.

    else,
        prob_in_FOV ← (%_time_looking_in_FOR/100) * FOV/FOR

If entity is not in the FOR,

if it recently fired its weapon and %_time_looking_in_FOR < 100,

    prob_in_FOV ← 1.0

else,

$$\text{prob\_in\_FOV} \leftarrow ((1 - \%\_\text{time\_looking\_in\_FOR})/100) * \text{FOV}/(2\pi - \text{FOR})$$

If prob_in_FOV > 0,

    if acquisition_level is none,
    factor ← acquisitionFactor(acqLevelBeforeBlinking ≥ Detection)

- acquisitionFactor as above.

    if (detection_bars * factor ≥ threshhold[][] * cycleN50Det)
        ratio = detection_bars * factor/cyclesN50Det
        If (ratio ≤1.8),
            W = 2.7 + (0.7 * ratio)
            pDetectInfinite(ratio) = ratio ** W/(1 + ratio ** W)
            factor ← pDetectInfinite(ratio)/3.4.
        Else,
            factor ← ratio/6.8.
        power = - mTimeOnTgt * factor
        pDetec(ratio, mTimeOnTgt) = 1.0 -exp(power)
        probability ← pModel → pDetec(ratio, mTimeOnTgt)
        probability ← probability * prob_in_FOV
        if (probability < draw),
            not acquired. Break to calculate acquisition level difference below.
        acquisition_level ← Detection

    if acquisition_level is Detection,
    factor = acquisitionFactor(acqLevelBeforeBlinking ≥ Classification)
    if (true_bars * factor ≥threshhold[][] * cycleN50Class)
        ratio ← true_bars * factor/cyclesN50Class
        probability ← pModel -> pDetec(ratio, mTimeOnTgt)
        probability ← probability * jumpiness
        if (probability < draw),
            no change in acquisition level. Break to calculate acquisition level difference below.
        acquisition_level ← Classification

    if acquisition_level is Classification,
    factor ← acquisitionFactor(acqLevelBeforeBlinking ≥ Recognition)
    if (true_bars * factor ≥ threshhold[][] * cycleN50Recog)
        ratio ← true_bars * factor/cyclesN50Recog
        probability ← pModel -> pDetec(ratio, mTimeOnTgt)
        probability ← probability * jumpiness
        if (probability < draw),

no change in acquisition level. Break to calculate acquisition level difference below.

acquisition_level ← Recognition

if identification at recognition

acquisition_level= = identification

break

if acquisition_level is Recognition,

factor ← acquisitionFactor(acqLevelBeforeBlinking ≥ Identification)

if (true_bars * factor ≥ threshhold[][] * cycleN50Ident)

ratio = true_bars * factor/cyclesN50Ident

probability ← pModel -> pDetec(ratio, mTimeOnTgt)

probability ← probability * jumpiness

if (probability < draw),

no change in acquisition level. Break to calculate acquisition level difference below.

acquisition_level ← Identification

change_in_acquisition_level ← acquisition_level - old_acquisition_level.

acquisition_priority ← old_acquisition_priority + 4.0 * change_in_acquisition_level.

Put the entity on the acquisition list for this sensor.

Once all entities have been scanned, sort the list by acquisition priority and trim it to the defined number of entities.

- The number of entries on a sensor's acquisition list is defined in Scenario Editor/Sensors/General tab.

Appendix 3

NVEOL Thermal Sensors

The algorithms in JCATS were derived from the Night Vision Electro-Optical Lab (NVEOL) model. **How do we differ?

At the start of the simulation a 128X128 matrix is generated from the NVEOL Detection Map used in JANUS(A) 5.0. The Detection Map consists of one hundred values representing a log normal distribution. JCATS randomly selects from the Detection Map while filling a 128X128 matrix. All viewer/entity pairs in the simulation are then hash mapped to the matrix. This means that for a given simulation run, a given viewer/entity pair will always have the same acquisition threshold value. However, due to the random fill of the matrix, the same viewer/entity pair may (and probably will) have a different threshold in subsequent runs.

Some terms that will be used in the following discussion are:

- threshhold[viewer][entity] is one of a hundred numbers representing a log normal distribution. It is applied to the cycles constants described below for the various levels of detection.
- cyclesN50Detection, cyclesN50Classification, etc., are the bars needed for a 50% probability of the corresponding level of acquisition given infinite time. They are:
- cyclesN50Detection ≡ 1.0
- cyclesN50Classification ≡ 2.0
- cyclesN50Recognition ≡ 3 .5
- cyclesN50Identification ≡ 6.4

NVEOL Thermal Sensor Scan

If the tests described in the General Sensor Scan section have been passed, proceed as follows:

If the viewer is under water, no acquisition by NVEOL sensor is possible. Exit.

Check LOS. If blocked, ignore the entity.

If entity to be acquired is within two meters of the sensor, consider it within the FOR.

If the entity to be acquired is not in the FOR and peripheral acquisition is off, ignore it.

Get NVEOL IR index for the entity

If the entity is in defilade

    Diveide the index by 2

End

Get ln(Delta T_at_target) from a table by the index found.

optical_size ← sqrt(optical_dimension * height(posture, defilade) * LOS_exposure_fraction) * transmission_factor

- optical_dimension comes from the PhysicalPropertyModel (DATA), and is different for humans versus all other entities.
- height is defined for non-human systems in Scenario Editor/Systems/Vehicle Data tab. For humans, height is 1.75 meters. In both cases it is adjusted for the entity's posture and defilade state.
- LOS_exposure_fraction is the fraction of total height to which the sensor has unobstructed LOS.
- transmission_factor is calculated using PLOSB through intermediate terrain features and smoke, if any.
- PLOSB is the probability that LOS is blocked per 10 meters of this terrain feature and is defined for a given type of terrain in the Terrain Editor.

If range <= 10 meters,

    optical_size ← 100 * optical_size

- range is the distance from the sensor to the entity to be acquired in meters. It is calculated in three dimensions.

ln(Delta T_at_sensor) ← ln(Delta T_at_target) - extinction*range – optical_len

- extinction (really the ln of it) is loss of contrast resulting from normal atmospheric effects. This value comes from the weather type entered in Scenario Editor/Tools/Scenario Parameters/Environment/Weather Conditions tab and is a function of range.
- Optical_len = 0

If (ln(Delta T _at_sensor) < ln(min Delta T))
    sensitivity(ln(Delta T _at_sensor)) ← 0.0.
Else if (ln(Delta T _at_sensor) > ln(max Delta T)),
    sensitivity(ln(Delta T _at_sensor)) ← maxCyclesPerMilliRadian.
Else,
    sensitivity(ln(Delta T _at_sensor)) ← value from slope, intercept calculation.
true_bars ← sensitivity(ln(Delta T _at_sensor)) * (optical_size /range) * 1000

- true_bars are the bars of resolution used to determine acquisition level.

If currentSimTime() < weaponsEffectEnd,
    WeaponsEnhancementMultiplier ← weaponsEffectMultiplier.
Else,
    WeaponsEnhancementMultiplier ← 1.0.
If speed > movingTargetSpeed,
    detFactor ← movingTargetSize.
Else,
    detFactor ← 1.0.
detection_bars ← true_bars * weaponEnhancementMultiplier * detFactor

- detection_bars are the bars of resolution used to test for detection.
- weaponEnhancementMultiplier accounts for the increased probability of detecting a system that just fired its weapon.
- detFactor increases the effective size of a moving system.

This is how to get the acquisition factor for any of the levels:
If the viewer just blinked (is suppressed),
    acquisitionFactor ← acquisitionFactor * reacquisitionFactor

- reacquisitionFactor is defined in Scenario Editor/Tools/Scenario Parameters/Human Factors/Acquisition tab.

If the viewer is moving (speed ≥ 0.25m/s),
    acquisitionFactor ← acquisitionFactor * movingSensorSize

- movingSensorSize is defined in Scenario Editor/Tools/Scenario Parameters/Human Factors/Acquisition tab.

If(detection_bars < threshhold[viewer][entity] * cyclesN50Det), ignore it.

- threshhold[][] is the value from the 128X128 matrix described earlier.

Else,
    acquire it (verify you have LOS).
If this is a new acquisition (not on the old acquisition list),
    acquisition_priority ← 0.5 * detection_bars for entities outside the FOR, or
    acquisition_priority ← 1.0 * detection_bars for entities inside the FOR.
If the entity is in the FOR,
    if it recently fired its weapon,
    prob_in_FOV ← 1.0

- Just Fired Time is defined in Scenario Editor/Tools/Scenario Parameters/Human Factors/Acquisition tab.

    else,
        prob_in_FOV ← (%_time_looking_in_FOR/100) * FOV/FOR
If entity is not in the FOR,
if it recently fired its weapon and %_time_looking_in_FOR < 100,
    prob_in_FOV ← 1.0
else,
    prob_in_FOV ← ((1 - %_time_looking_in_FOR)/100) * FOV/(2π - FOR)

If prob_in_FOV > 0,
    if acquisition_level is none,
    factor ← acquisitionFactor(acqLevelBeforeBlinking ≥ Detection)

- acquisitionFactor as above.

```
        if (detection_bars * factor ≥ threshhold[][] * cycleN50Det)
                    ratio = detection_bars * factor/cyclesN50Det
                    If (ratio ≤1.8),
                        W = 2.7 + (0.7 * ratio)
                        pDetectInfinite(ratio) = ratio ** W/(1 + ratio ** W)
                        factor ← pDetectInfinite(ratio)/3.4.
                    Else,
                        factor ← ratio/6.8.
                    power = - mTimeOnTgt * factor
                    pDetec(ratio, mTimeOnTgt) = 1.0 -exp(power)
                    probability ← pModel → pDetec(ratio, mTimeOnTgt)
                    probability ← probability * prob_in_FOV
                    if (probability < draw),
                        not acquired. Break to calculate acquisition level difference below.
                    acquisition_level ← Detection
        if acquisition_level is Detection,
        factor = acquisitionFactor(acqLevelBeforeBlinking ≥ Classification)
        if (true_bars * factor ≥threshhold[][] * cycleN50Class)
                    ratio ← true_bars * factor/cyclesN50Class
                    probability ← pModel -> pDetec(ratio, mTimeOnTgt)
                    probability ← probability * jumpiness
                    if (probability < draw),
                        no change in acquisition level. Break to calculate acquisition level
                        difference below.
                    acquisition_level ← Classification
        if acquisition_level is Classification,
        factor ← acquisitionFactor(acqLevelBeforeBlinking ≥ Recognition)
        if (true_bars * factor ≥ threshhold[][] * cycleN50Recog)
                    ratio ← true_bars * factor/cyclesN50Recog
                    probability ← pModel -> pDetec(ratio, mTimeOnTgt)
                    probability ← probability * jumpiness
                    if (probability < draw),
                        no change in acquisition level. Break to calculate acquisition level
                        difference below.
                    acquisition_level ← Recognition
        if identification at recognition
                acquisition_level= = identification
                break
        if acquisition_level is Recognition,
                    if silhouetted
                        break
                    factor ← acquisitionFactor(acqLevelBeforeBlinking ≥ Identification)
        if (true_bars * factor ≥ threshhold[][] * cycleN50Ident)
                    ratio = true_bars * factor/cyclesN50Ident
                    probability ← pModel -> pDetec(ratio, mTimeOnTgt)
                    probability ← probability * jumpiness
                    if (probability < draw),
                        no change in acquisition level. Break to calculate acquisition level
                        difference below.
                    acquisition_level ← Identification
change_in_acquisition_level ← acquisition_level - old_acquisition_level.
If change_in_acquisition_level  > 0
        acquisition_priority ← old_acquisition_priority + 4.0 * change_in_acquisition_level.
End if
Put the entity on the acquisition list for this sensor.
Once all entities have been scanned, sort the list by acquisition priority and trim it to the defined number of entities.
```

- The number of entries on a sensor's acquisition list is defined in Scenario Editor/Sensors/General tab.

Appendix 4

Algorithm number 21

Planned Indirect Fire

Target line (manually entered)
Who is shooting (one or more)
The line is divided equally to the number of shooters, each shoots at the center of its piece.
Munition
Mission type (ASAP, priority, timed)
Number of volleys

I. Schedule mission:

```
 Loop over all potential shooters
       Sum number of target points
End
If number = 0  can't schedule
Else
     Calculate target points (divide line to number of shooters and find center
     of each)
End if
Loop over all shooters
       Assign target points to each
End
```

- Number of points is in NumberOfArtlleryTubes

```
       if we are operational
              loop over all weapon stations
                     if the station can do
                            return 1
                            break
                     else
                     return 0
                     end if
              end loop
       else
       return 0
```

- To find if a station can do:

```
if I am dead
   can't do
else
loop over all weapon stations
       pick ammo for the request
       if possible
              can do
```

```
        else
        can't do
        end if
    end loop
    end if
```

- How to pick ammo for request:

```
        Make sure not broken and can fire in  indirect mode
        If we selected one and it's not me
                can't fire
        endif
        Loop over all munitions
                If the munition is useable for artillery
                        Return 1
                Else
                        Return 0
                Endif
        end
```

- How to find out if munition is useable

```
        if there is a selected munition and I am not the one or there is no selection
                and can be fired in indirect mode and I am the right type
                        make sure I am not sensor guided
                        make sure I am not crew guided
                        make sure I am not self guided
                        return true
        else
                        return false
        endif
```

- How to assign target points

```
        If I am not operational
                don't schedule
        else
                loop over all weapon stations
                        try to schedule mission
                        stop on the first chance
                        break
                end
        endif
```

- How to schedule mission

```
        If I am dead – can't schedule
        If no more points left – can't schedule
        Loop over all my weapons
                Pick ammo
                Make artillery mission
                Queue it
                Return true
        End
```

II. Start Artillery:

```
    If no mission return false
    If the first in line is active return true (don't start another)
    If mission should start now
        return false
    else
        create artillery engagement
        start engagement
        return true
    endif
```

- How to create artillery engagement

```
Set mission to active
Find time we can shoot (out of defilade)
Find time we can fire (load)
if when_to_fire < 0
    can't do
    break
Time_to_fire = maxc (Time_to_fire, Time_to_shoot)
If not ASAP mission
    If time to start shooting < Time_to_fire
        Abort
    Else
        Time_to_fire = Time_first_volley
    End
Calculate range
Calculate number of volleys
```

III. Shoot Artillery:

```
Find our position (x,y coordinates)
Get target position (x,y coordinates)
• Take your piece of the line, divide by the number of volleys and shoot at the
  center of each.
If it is a grenade see later what to do in this case
If this is the first volley
    Calculate aiming error
Endif
Check range to target
If munition range < target range
    Abort mission
Endif
Calculate aiming point based on aiming error
• aiming point = target position + aiming error (set z coordinate to 0)
If this is a grenade we need to correct for the proper floor
Shoot at the aiming point
If we didn't get a shot
    Abort mission
    Decriment the number of volleys
        If number of volleys = 0
            Done
            Stop engagement
        Endif
Make weapon ready to fire with that munition
If it can't be made ready
    Abort engagement
Endif
Queue event
```

- How to calculate aiming error

```
If FASCAM and not grenade
    Aiming error = 0
Else
    Find indirect fire aiming error (next bullet)
endif
```

- How to find indirect fire aiming error
  Given launch point and aim point

Range = distance from launch to aim
If range ≤ 2
    Aim error = 0
Else
    Find a unit vector in the direction of shot
    Find a unit vector perpendicular to it
    Look up the indirect fire range table for the ammunition
    • For each range the table contains: Time of flight
        Angle of fall
        Aim error in 2 directions
        Ballistic error in 2 directions
    Interpolate (linearly) based on range
    • keep it constant outside range
    error = range error * normally distributed random number +
        deflection error * normally distributed random number

• What to do in the case of grenades
    Find my_floor (environmental model if in building and what floor)
    If my_floor $\neq 0$
        If shooter is in the same playbox as target (exclude tunnels)
            Target position z coordinate = shooter z coordinate
        Endif
    Endif
    Check throwing the grenade (allow 1 meter to either side for side-arm throwing
    Check if grenade is blocked where you are or where the 1 meter can throw

• How to find out if grenade is blocked
    If line from shooter to target is blocked or there are systems in the way
        return true (blocked)
    endif
    Loop on seven different angles from horizontal
        Construct a parabola from shooter to target with that angle
        If the parabola is not blocked
            Return false
        Endif
    End loop

IV. Impact Point:
    Given launch point and aim point (with aiming error)
    If range ≤ 2
        Aim point = impact point
    Else
        Find a unit vector in the direction of shot
        Find a unit vector perpendicular to it
        Look up the indirect fire range table for the ammunition
        Interpolate (linearly) based on range
        • keep it constant outside range
        ballistic error = range error * normally distributed random number +
            deflection error * normally distributed random number
        Compute impact point (including ballistic error)
        • This gives z coordinates based on terrain
        • In the case of grenade – correct for height
          z impact point = airburst height (specified) + z impact point
        Check if the round is blocked on the way down to impact
        • Compute angle of fall from impact based on the range table
          In case of a bomb drop the angle is 90 degrees

```
    Take a unit vector in this direction
    Multiply the unit vector by min (1000 m, 25% range)
    Calculate that point and check if projectile is blocked
  If it is blocked
        Impact point = point of blockage – 5 cm
  Endif
Endif
```

Appendix 5

Algorithm number 20

Planned Direct Fire

Given target position center and radius
Pick a list of shooters

```
I.      Schedule direct fire
        If not operational
            done
        else
        loop over all weapon stations
            if weapon station can schedule mission
                done
        end

        • How to schedule a mission
            If dead or blind (no sensor) return false
            If direct fire at entity and not acquired by our sensor return false
            If can find direct fire munition weapon pair
                  • first for beam weapon then for other weapons
                Create a mission for direct fire
                Create a mission for beam weapon (target is picked differently)
                  • see later
                Queue mission
                Return true
            Else
                Return false
        • How to find munition weapon pair
            Loop over all weapons
                If can direct fire and (this is ASAP or
                    time for direct fire setup < required time to first shot)
                        Tell weapon to pick direct fire ammunition
                        If it can
                            if the suppression of that weapon is
                                better than the best so far
                                    make this best weapon
                            endif
                endif
            end
          if we have best weapon return true

        • How to find if a weapon can direct fire
            If not broken and can be used in planned direct and
```

```
                (no selection or I am selected)
                        Loop over all ammunitions
                                If can fire direct
                                        Return true
                                        Break
                                Endif
        Return false
```

• How to pick direct fire ammunition

```
        If request to use beam weapon and I am not beam weapon
                Can't pick
        Else
                If request not beam weapon and I am beam weapon
                        Can't pick
        Endif
        If not broken and can be used for direct fire
                Loop over all munitions
                        Get the direct fire suppression indicator (ind)
                        If ind > 0
                                If request is beam
                                        Take this munition
                                        Break
                                Else
                                        Ratio = Suppression indicator/sustained
                                                cycle time
                                        if this is best so far
                                                take this ratio as best so far
                                        endif
                                endif
                        endif
                end loop
        endif
```

II. Start direct fire

```
        If no mission return false
        If mission in front is active return false
        If mission in front should not start now return false
        Start mission
                If mission can be started
                        Create direct fire engagement
                        Break
                Endif
                If we don't have an engagement return false
        Start direct fire engagement
        Return true
```

• How to create direct fire engagement

```
        If direct fire at target
                Create and return direct fire at target engagement
        Else
                Create and return direct fire at area engagement
        Endif
```

• How to start direct fire engagement

This is given in two parts. Part A for area and part B for target

A. Find Time when to fire

```
        If time when to fire < 0
                Stop
```

```
            Find time when can shoot (clear defilade)
            Time when to fire = max of the  two times
            If timed mission and time of first shot is before we can fire
                Abort
            Else
                Time to fire = time of first shot
            Endif
        B.  If we can't see the target
                Abort
            Find time when weapon is ready to shoot
                If time < 0 or target is dead
                        Abort
          Find time when weapon is ready to shoot out of defilade
           Time to shoot is the max of the two
           If timed mission
                If we can't shoot in time
                        Abort
                Else
                        Time to shoot = time of first shot
                Endif
           Endif
```

```
III.    Shoot direct fire
      This is given in two parts. Part A for area and part B for target
        A.  If center of area is NOT in range
                  Abort
         Else
                  Pick a target position in the area at random
                  Fire
                    If weapon doesn't fire
                            Abort
                    Else
                    Cycle weapon and ask when it is ready
                    Endif
                    If mission is over (time is up)
                            Done
                    Endif
                    If weapon is not ready (broken or out of ammo)
                            Done
                    Endif
            Endif
        B.  If target is dead
                  Abort
            Endif
            If target is out of range
                    Abort
            Endif
            If LOS is lost
                    Abort
            Endif
            Pull the trigger
            If failed
                    Abort
            Else
                    Cycle weapon and reload if necessary
                    If mission time is up
                            Stop engagement
```

```
            Endif
            If weapon is not ready
                    Stop engagement
            Endif
    endif
```

- How to pick a target at area

```
    If area is in building
            Change the area to vertical about the diameter
             perpendicular to line of shooter (keep the same floor)
    else
            Drape circle to terrain
    endif
```

- How to pick a target for beam weapon

    Doesn't shoot randomly but sweeps across the circle
    aiming 1 meter above terrain
    The sweep is from left to right along the diameter
    Perpendicular to line from shooter to center of target area
    The step size is the beam diameter at range

Appendix 6

Beam Weapons

The weapon category used to define a directed-energy system is the beam weapon. The munition of a beam weapon is described as a pulse length, i.e., pulses of 1, 2, and 3 seconds describe three different munitions.

Effects

The incapacitation (suppressive) effects of each munition (pulse length) against each vulnerability category is given in the table associated with that category and the beam weapon. An example of such a table is shown in Figure 1.

Suppression Degradations

Range (m)	Speed	Position Prep	Shoot PH	Shoot Prep	Acq	Rest	Energy Loss	Energy StDev	Supp Time	Supp StDev
0.00	0.10	6.00	0.10	6.00	6.00	0.10	2000.	2.00	120.0	2.00
50.00	0.20	5.00	0.20	5.00	5.00	0.20	1000.	1.50	90.00	1.50
100.00	0.30	4.00	0.30	4.00	4.00	0.40	500.	1.00	60.00	1.00
200.00	0.40	3.00	0.40	3.00	3.00	0.60	250.	0.75	30.00	0.75
300.00	0.50	2.00	0.50	2.00	2.00	0.80	100.	0.50	10.00	0.50
400.00	1.00	1.00	1.00	1.00	1.00	1.00	50.0	0.00	0.00	0.00
0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00

Figure 1. Data table for suppressive effects of beam munitions.

Each value in the table describes, as a function of range from the weapon to the target, a degradation to the target's ability to perform actions after being struck by the beam weapon. These values are multipliers for the parameters described by the column headings. For example, using Figure 1, a target shot at a range of 100 m would have:

—his speed reduced to 30% of his normal speed
—his time to prepare a position increased by a factor of 4
—his shoot PH degraded to 30% of its normal value
—his shoot preparation time increased by a factor of 4

—his acquisition times increased by a factor of 4
—the value of rest reduced to 40% of its usual effect
—an energy loss averaging 500 energy units,
    with a standard deviation of 1 energy unit (normally distributed)
—these effects last for an average of 60 seconds,
    with a standard deviation of 1 second (normally distributed).

Weapon description

Parameters needed to define the beam weapon are:

Minimum range
Maximum range
Setup time
Lay time
Lay time per 90 degrees
Tear down time
Duty cycle
Range parameters

Minimum and maximum range define the minimum and maximum ranges at which the system can be used.

Setup time refers to the time required to get the weapon ready to fire after moving it or turning it off.

Lay time is the time needed for the shooter to aim at his target.

Lay time per 90 degrees is not currently implemented. In future versions of JCATS this parameter will be used to define how long it takes to re-aim through an angle. This can be thought of as the time required to slew the weapon; the lay time will then describe the finer adjustments needed to aim the weapon.

Tear down time is the opposite of setup time and refers to the time required to prepare the weapon for movement or to turn it off.

Duty cycle is given in percent. As currently implemented, the beam weapon will fire one shot, the needs to recover for the amount of time defined by the duty cycle. The recovery time is given by:

$$\text{recovery time} = \frac{\text{pulse length} * (1 - \text{duty cycle})}{\text{duty cycle}}$$

For example, if the duty cycle is 10% and the pulse length is 2 seconds, then the weapon will have to recover for 18 seconds (2 sec * 0.9 / 0.1).

The range parameters of the weapon are described in a table consisting of three columns: range, beam diameter, and pulse length. At ranges from the minimum weapon range to the first entry in the table, the first row of the table is used. At ranges from the last table entry to the maximum range of the weapon, the last row of the table is used. Between these, data for beam diameter are interpolated. At a given range, the pulse length used is that corresponding to the nearest table entry for range.

As an example, consider a weapon with a 10 m minimum range, a 500 m maximum range, and range parameters as follow:

Range (m)	Beam diameter (m)	Pulse length (sec)
100	1.0	1.0
200	1.5	3.0
400	3.0	5.0

At ranges from 10 to 100 m, the beam diameter is 1 m and a 1 second pulse is used. At ranges between 100 and 400 m, values for beam diameter are interpolated from the table. From 400 to 500 m, beam diameter is 3.0 m and pulse length is 5.0 sec.

Using the same table, at ranges from 10 - 150 m, a 1-sec pulse is used. From 150 - 300 m, a 3-sec pulse is used. Beyond 300 m, a 5-sec pulse is used.

N.B. The user must take care to ensure that every pulse length referenced in the range parameter table corresponds to a munition of the same pulse length.

As currently implemented, the only mode of use for a beam weapon is planned direct fire. The user defines a circle that is his target. After the lay time, the beam weapon will begin to sweep across the circle from one edge to the other. The distance from the weapon to the target determines the beam diameter, and the weapon will shoot one beam diameter, then, after the recovery time defined by the duty cycle, the weapon will move over by that diameter and fire the next pulse, etc., until the entire circle has been traversed.

Appendix 7

Algorithm Number 6
Enhanced Lighting

Parameters:
1. CosViewingAngle =cos (88 deg) = 0.0349
cosine of the smallest angle between a panel normal and the sensor-to-panel vector at which the sensor can still reasonable see the panel.

2. CosLightNearTarget =cos(2.5 deg.)= 0.99905
cosine of the largest angle at which the subject is considered backlit/silhouetted

3. SinAboveTheHorizon =sin(1 deg.)= 1.7452e-2
sine of the angle measured positive vertically above the horizon above which the target is assumed to be seen against the sky. Below this angle, the target is assumed to be seen against the ground

4. SteradiansOfSunMoon = 6.5e-5
The solid angle subtended by the Sun and the Moon

5. CosMaxScattering =cos(5 deg)= 0.99619
The cosine of the maximum angle at which forward scattering of lights into the sensor occurs.

6. ScatteringFraction = 0.02
The fraction of light within the forward scattering cone (as defined by CosMaxScattering above) that forward scatters into the sensor

7. AreaLightSourceFraction = 0.1
The fraction of an area light (the sensor is in, but the target is not) that enters the sensors and overlays both the target and the background.

minLightLux = sensorMinLux / 5.0
compute unit vector from sensor to top of target
compute unit vector from sensor to bottom of target
compute cosBetweenTopAndSensor = cosine of angle between normal to top surface of target and vector from sensor to target top
topVisible if cosBetweenTopAndSensor ≥ CosViewingAngle

compute cosine of angle between front (facing sensor) surface and vector from sensor to target
frontVisible if this cosine ≥ CosViewingAngle

compute cosBetweenSideAndSensor = cosine of angle side normal and vector from sensor to target
If this cosine is negative make normal point the opposite direction
sideVisible if this cosine ≥ CosViewingAngle

facetNormal = unit vector sum of(frontNormal+ sideNormal + topNormal)
facetVisible if cosine between this vector and sensor to target vector ≥ CosViewingAngle

facet2Normal = unit vector sum of(frontNormal - sideNormal + topNormal)
if cosine between facet2Normal and sensor to target vector < 0.0
  take facet2Normal = unit vector sum of(sideNormal - frontNormal + topNormal)

facet2Visible if cosine between facet2Normal and sensor to target vector ≥ CosViewingAngle

STEP 0: Initialize the luminance levels
luxOfTop = 0.0
luxOfFront = 0.0
luxOfSide = 0.0
luxOfFacet = 0.0
luxOfFacet2 = 0.0
luxOfBackground = 0.0
luxScattered = 0.0

STEP 1: Compute the contribution due to the NATURAL LIGHT SOURCES
  (i.e. Sun/Moon) direct and sky/ground reflected light

Get skyLux and grndReflectivity (input)
luxOfSide = skyLux *((1.0 + grndReflectivity) / 2.0) * targetReflectivity

luxOfFront = skyLux *((1.0 + grndReflectivity) / 2.0) * targetReflectivity

luxOfTop = skyLux * targetReflectivity, if topVisible and 0 otherwise

luxOfFacet2 = skyLux * (3.0 + grndReflectivity) / 4.0) * targetReflectivity

luxOfFacet = skyLux *((3.0 + grndReflectivity) / 4.0) * targetReflectivity

luxOfBackground = skyLux if sensorToTarget.z > SinAboveTheHorizon
  skyLux / skyToGroundRatio otherwise

Compute direct illumination contribution as follows:
get lightLux = illumination of mNaturalBackGroundLight
 if lightLux ≥ minLightLux and lightLux > 0.0 and elevation of mNaturalBackGroundLight ≥ 0.0
  MaxEarthTerrainHeight = 9200.0
 Find unit vector from the target to the Sun/Moon as
 fromSunOrMoon=(-cosPhi*cos(theta),
  -cosPhi*sin(theta),
  -sin(elevation of mNaturalBackGroundLight)
 where cosPhi = cos(elevation of mNaturalBackGroundLight)
  theta = π/2.0 - azimuth of mNaturalBackGroundLight
 Make sure Sun/Moon shining down
  if fromSunOrMoon.z > 0.0 fromSunOrMoon.z = 0.0

  if targetPosition.z < MaxEarthTerrainHeight
   Calculate minimum distance we have to go back towards sun/moon
    Calculate sunPosn = position of Sun/Moon
   subtract minimum distance and add target position
   Run line of sight from light source (sunPosn) to target position and get transmissionFraction and
    exposureFraction

```
        else
      transmissionFraction =  exposureFraction = 1
   endif
 Attenuate  light due to partial transmission
   lightLux =lightLux* transmissionFraction

 Add direct attenuated Sun/Moon backlight if behind target
   Compute  cosSunToObserverTarget = -cosine of angle between vectors fromSunOrMoon and
                                                   sensorToTarget
   if cosSunToObserverTarget ≥ CosLightNearTarget
          luxOfBackground =luxOfBackground+
               cosSunToObserverTarget * lightLux / SteradiansOfSunMoon


 Determine the lux reflected from each panel
   lightLux =lightLux /π
   lambertCos = cosine of the angle between normal to panel and  fromSunOrMoon
   if lambertCos > 0.0
          luxOfTop = luxOfTop +lambertCos * lightLux * targetReflectivity
          luxOfBackground =luxOfBackground + lambertCos * lightLux * grndReflectivity
   endif

   All the remaining panels are target panels so multiply in  reflectivity
   lightLux = lightLux * targetReflectivity

 Assume exposed portion is on top, so sides have less light
   lightLux =lightLux * exposureFraction

 Compute the Front Surface contribution
        if frontVisible
      lambertCos = cosine angle between frontNormal and  fromSunOrMoon
          if lambertCos > 0.0
           luxOfFront =luxOfFront + lambertCos * lightLux
   endif
 Compute the Side Surface contribution
   if sideVisible
      lambertCos = cosine angle between sideNormal and  fromSunOrMoon
          if lambertCos > 0.0
           luxOfSide = luxOfSide + lambertCos * lightLux
   endif
 Compute the Facet Surface contribution
   if facetVisible
      lambertCos = cosine angle between facetNormal and fromSunOrMoon
          if lambertCos > 0.0
           luxOfFacet =luxOfFacet + lambertCos * lightLux
   endif
 Compute the Facet2 Surface contribution
   if facet2Visible
      lambertCos = cosine angle between facet2Normal and fromSunOrMoon
          if lambertCos > 0.0
           luxOfFacet2 =luxOfFacet2 + lambertCos * lightLux
   endif
```

STEP 2: Compute how much the spot and area light shine on the TARGET.

For each light source call lightOnTarget (see below) to compute:
lightLux and insideLight indicator

```
if insideLight
      luxOfBackground = luxOfBackground + grndReflectivity * lightLux
      lightLux = lightLux * targetReflectivity
      luxOfTop = luxOfTop + lightLux
      luxOfFront =luxOfFront + lightLux
      luxOfSide = luxOfSide + lightLux
      luxOfFacet =luxOfFacet + lightLux
      luxOfFacet2 =luxOfFacet2 + lightLux
else
  compute unitVectorFromLightToTarget
       lambertCos = cosine angle between normal to XYPlane and unitVectorLightToTarget
       if lambertCos > 0
        luxOfBackground =luxOfBackground + lambertCos * lightLux * grndReflectivity
        if topVisible
          luxOfTop = luxOfTop + lambertCos * lightLux * targetReflectivity
        endif
  endif

         Get target reflection intensity
       lightLux = lightLux *targetReflectivity* targetExposureFraction
  Compute the Front Surface contribution
       if frontVisible
        lambertCos = cosine angle between frontNormal and unitVectorLightToTarget
        if lambertCos > 0.0
          luxOfFront = luxOfFront + lambertCos * lightLux
        endif
  endif
  Compute the Side Surface contribution
       if sideVisible
     lambertCos = cosine angle between sideNormal and unitVectorLightToTarget
         if lambertCos > 0.0
          luxOfSide = luxOfSide + lambertCos * lightLux
         endif
  endif
  Compute the Facet Surface contribution
       if facetVisible
         lambertCos = cosine angle between facetNormal and unitVectorLightToTarget
         if lambertCos > 0.0
           luxOfFacet = luxOfFacet + lambertCos * lightLux
         endif
  endif
  Compute the Facet2 Surface contribution
       if facet2Visible
         lambertCos = cosine angle between facet2Normal and  unitVectorLightToTarget
         if lambertCos > 0.0
           luxOfFacet2 = luxOfFacet2 + lambertCos * lightLux
         endif
  endif
```

STEP 3: doesn't exist in code
STEP 4: Determine which spotlights and area lights shine into the sensor to aid in silhouetting the target

```
For each light container:
   call lightOnSensor (see below) to get thisLightsLuxScattered and  thisLightsLuxBackground

   luxOfBackground = luxOfBackground + thisLightsLuxBackground
   luxScattered = luxScattered + thisLightsLuxScattered
end loop
```

```
Compute the contrast of the TARGET as follows
 set: computedContrast = 0.0
   isTargetSilhouetted = true
 minLux = sensorMinLux
 if minLux ≤ 0.0 minLux = 1.0e-6
 if topVisible
  call contrast to get  computedContrast and isTargetSilhouetted
 endif
 if frontVisible
  call contrast to get  computedContrast and isTargetSilhouetted
 endif
   if sideVisible
  call contrast to get  computedContrast and isTargetSilhouetted
 endif
 if facetVisible
  call contrast to get  computedContrast and isTargetSilhouetted
 endif
 if facet2Visible
  call contrast to get  computedContrast and isTargetSilhouetted
 endif

 Compute the log of the contrast
 computedContrast = -1.0e20                    if computedContrast = 0.0
                    log(computedContrast) otherwise
```

Contrast:

Contrast calculation and choosing

Parameters: SilhouetteRatio = 64.0
In order to be silhouetted, the background/target ratio must be more than this.

```
s = true  if luxTarget < minLux
   false otherwise
if luxTarget > maxLux
    luxTarget = maxLux
    s = true
else if luxTarget < minLux
      luxTarget = minLux
      s = true
endif
if luxBkgd > maxLux
     luxBkgd = maxLux
     s = true
else if luxBkgd < minLux
      luxBkgd = minLux
endif

Calculate numerator of contrast ratio
num = luxTarget - luxBkgd
if num < 0.  num = -num
if not s and  (luxTarget ≤ 0.0 or  luxBkgd/luxTarget > SilhouetteRatio)  s = true
c = num / (luxBkgd + luxScattered)
```

Choose between existing values and new ones. Choose unsilhouetted contrast when available
if (not s and silhouetted and c > 0.0) or (not s or silhouetted) and c > contrast)
    contrast = c
    silhouetted = s
endif

LightOnTarget (For spot light, flare and area light)
Initialize the exposure fraction
exposureFraction = 1.0
Call lightOnPosition (see below) to get lightLux and insideLight
targetExposureFraction = exposureFraction
return lightLux

lightOnSensor (For spot light and flare)
Initialize:
luxScattered=0.0
luxBackground = 0.0

Get the unit vector from light to sensor
compute cosLightVsTarget =cosine angle between vectors from sensor to light and to target
  if cosLightVsTarget < CosMaxScattering return
Call lightOnPosition to get light Lux and insideLight
  lightLux = lightLux *exposureFraction * π
  if cosLightVsTarget > CosLightNearTarget
    luxBackground = luxBackground +lightLux
  else
    luxBackground = luxBackground +ScatteringFraction * lightLux
    luxScattered = luxScattered +ScatteringFraction * lightLux
  endif
  return

LightOnSensor (for area light)
Initialize
    luxScattered = 0.0
  luxBackground = 0.0

  if light is turned off return
  if sensorPosition.z ≤ mCenterTopOfLight.z and we are inside light
    if target is inside light
        luxScattered = luxScattered +AreaLightSourceFraction * mLuxesInLight
    else
      both sensor and target inside the light no scattering or background light is added
        return
    endif
  else
    Sensor not in the light
    compute unit vector sensorToLightCenter
    See if lit area is in front of sensor. If not, quit now, i.e.
    compute cosALightVsTarget = cosine angle between vectors from sensor to light center and to target
    if cosALightVsTarget ≤ 0.0 return
    Compute lPosn=the center of the area light

```
    if area light behind the target
      take lPosn as point of intersection behind target
      backLit = true
    else if light lies between sensor and target
         take lPosn as point of intersection behind target
         lightBetweenSensorAndTarget = true
            else if target is in light
           take lPosn as point of intersection behind target
           else
            error message
            return
           endif
        endif
     endif
  lightLux = AreaLightSourceFraction * mLuxesInLight
 if lightLux < 0.2 * minLux return
Compute line of sight from  lit area to sensor and get transmissionFraction and exposureFraction
   lightLux = lightLux *transmissionFraction * exposureFraction
   if backLit
         luxBackground = luxBackground +lightLux
         luxScattered = luxScattered + ScatteringFraction * lightLux
         return
   else if lightBetweenSensorAndTarget
       luxBackground = 0
            luxScattered = luxScattered + lightLux
       return
      else
        Estimate sensorToEdge = sensor to edge of light
        compute  l = norm of the vector sensorToEdge
         if l ≤ 0.0 or cosine angle between vectors from  sensor to edge and to target < CosMaxScattering
               return
          endif
        luxBackground = 0
        luxScattered = luxScattered +  ScatteringFraction * lightLux
      endif
    endif
```

LightOnPosition (for flare)

```
We are never "inside" a flare
  insideLight = false
  if  flare is out return 0.0
  illum = illuminance of targetPosition (see below)
  if illum ≤ 0.2*minLux return 0.0
Run a line of sight from light to position to find how much light is lost due to transmission
 Reduce the amount of light from flare by transmission fraction
  illum = illum *transmissionFraction
  return illum
```

lightOnPosition (for Spot)

```
We are never inside a spot light
  insideLight = false
```

```
if  Light is turned off  return 0.0
 illum = illuminance of targetPosition (see below)
if illum ≤ 0.2*minLux return 0.0
Run a line of sight from light to position to find how much light is lost due to transmission
Reduce the amount of light from spotlight by transmission fraction
 illum = illum *transmissionFraction / π
return illum
```

LightOnPosition (for area light)

```
Initialize
  insideLight = false
  if light is turned off  return 0.0
   if the point is inside the light
     insideLight = true
     return mLuxesInLight
   endif
   lightLux = mLuxesInLight
 Find distance outside of lit area.
For now lets use the approximate radius of the area light to subtract off the distance of the target from the light.
Compute lPosn = position from center to top of light
 Estimate distance from light perimeter
 If the entity is closer to the edge of the light than the lights  "radius" treat it like it is in the light modulo the source
fraction
  if lightDistSquared / lightRadiusSquared ≤ 4.0
    lightLux =lightLux * AreaLightSourceFraction * π
  else if lightDistSquared > 16.0 * lightRadiusSquared
       lightLux = AreaLightSourceFraction *  (2.0 * mHeight * mLightRadius) / lightDistSquared
     else
       compute approxLightDistToEdge
       Estimate subtended angle of light in XY plane (deltaTheta)
         deltaTheta = 2.0 * arc tan (mLightRadius,approxLightDistToEdge)
       Multiply lux of source by angle it the solid angle it subtends to get lumens on target.
       lightLux = AreaLightSourceFraction * mLuxesInLight * deltaTheta *
             sqrt(hSq/(hSq + (approxLightDistToEdge * approxLightDistToEdge)))
       where      hSq = mHeight * mHeight
     endif
 endif
 if lightLux ≤ 0.2 * minLux  return 0.0
  Perform LOS calculation from light to target
  lightLux =lightLux * transmissionFraction  / π
 return lightLux
```

illuminance(for Flare)

computes illuminance by flare at a given position

```
Ifthe position is outside the cone return 0.0
illum = 0.0
compute  distSquaredFromLight
  if distSquaredFromLight > 1.0
    illum = (mLumensOfLightInCone / distSquaredFromLight) * 100
  else
```

```
    illum = mLumensOfLightInCone
  return illum
```

illuminance (for Spot)

```
If the point is outside the cone  return 0.0
illum = 0.0
Compute  distSquaredFromLight
  if distSquaredFromLight > 1.0
    illum = mLumensOfLightInCone / distSquaredFromLight
  else
    illum = mLumensOfLightInCone
  return illum
```

Appendix 8

Algorithm Number 17
FASCAM

FASCAM, a FAmily of SCAttered Mines, comes in two classes, anti-tank and anti-personnel. Laying mines is done like artillery (see planned indirect fire, algorithm 21) with one difference. In FASCAM the aiming error and ballistic error are both zero. Adjudication of mines is explained in Encountering a Minefield (algorithm 30)

Appendix 9

Algorithm Number 28
Fatigue Factor

Fatigue factor is a degradation factor on a requested speed not max speed. It is done by a table look-up. The following is a table of movement speed factor based on energy level.

Movement speed factor	Energy level
1	81-100
.5	61-80
.4	41-60
.3	21-40
.2	0-20

Appendix 10: MOUT JCATS V&V-- Algorithm Upgrade

Background.

The Naval Postgraduate School (NPS) is participating in the MOUT JCATS Verification and Validation (V&V). Although the tasking has formally focused on verification of the JCATS algorithms, research by the author has revealed that the most appropriate algorithms were not used in JCATS in the first place. JCATS has expressed interest in exploring this point further, especially as regards accreditation of JCATS for MOUT studies.

The state of military modeling and simulation was quite different when Janus (from which JCATS has descended) was initially developed. The major difference (as pertains the V&V of MOUT JCATS) is that there was no attempt at model standards by the U.S. Army. Moreover, the development of model standards (e.g. standardization of algorithms) has also been accompanied by the development of compendia of algorithms by the U.S. Army for various reasons. Thus, there is information (detailed enough for a contractor to implement algorithms, including input data) now available on a number of algorithms.

Need for Upgrade.

"The Compendium of Close Combat Tactical Trainer Algorithms, Data, Data Structures, and Generic System Mappings" (AMSAA [1996a]) contains a number of algorithms appropriate for a high-resolution Monte-Carlo simulation like JCATS. These algorithms represent the best that U.S. Army weapon-system analysis has developed (e.g. see DARCOM [1977]). The author's own teaching and personal research at the Naval Postgraduate School (NPS) substantiates this assertion. For example, the AMSAA [1996] compendium of algorithms for the close combat tactical trainer (CCTT) does not employ the assumption, in general, of statistical independence between rounds because fire control usually introduces serial correlation between rounds. Moreover, AMSAA can supply input data that allows one to play such serial correlation in a high-resolution Monte-Carlo simulation like Janus or JCATS. The author's personal research has revealed that when such serial correlation exists (and is appreciable), significantly different results are obtained when one ignores such serial correlation by assuming statistical independence between rounds.

Moreover, the Army has apparently not kept Lawrence Livermore National Laboratory (LLNL), the developer of both Janus and also JCATS, explicitly informed about Army model standards and the significance of various compendia of high-resolution-simulation algorithms (Uzelac [2000]). Consequently, LLNL has not been aware that the latest (and most appropriate) algorithms were not being used in JCATS. Furthermore, the author has noted that even the Army version of Janus contains a direct-fire attrition algorithm that should be upgraded because independent rounds has been assumed (Taylor [1999a], [1999b]).

Sources of Information.

The U.S. Army has put together several compendia of algorithms for high-resolution Monte-Carlo simulations. The AMSAA compendium of algorithms for the CCTT (AMSAA [1996a]) has been noted above. This compendium has been subsequently updated (AMSAA [1999]). Additionally the Army has also developed a compendium for high-resolution attrition algorithms (AMSAA [1996b]) (see ODUSOR & AMSO [1997]). Further information about such compendia and Army model standards may be found "Army Model and Simulation Standards Report"s for various FYs. Lack of time has not allowed such sources to be thoroughly researched at this time.

Algorithms Requiring Upgrading. Preliminary research has revealed that the following algorithms need upgrading in JCATS:

(1) target acquisition,

(2) indirect-fire attrition,

(3) direct-fire attrition.

The order given above corresponds to their priorities, i.e. the first algorithm (target acquisition) has the highest priority. In particular, the ACQUIRE algorithm (two-dimensional target) should replace the obsolete Night Vision Laboratory (NVL) methodology (one target dimension). Moreover, the Army has apparently implemented the ACQUIRE in CASTFOREM (and other Army simulations) somewhat differently than LLNL has for the NVL methodology. Lack of time has prevented documentation of the details here.

Appendix 11: Flaw in Janus Direct-Fire Assessments

The following explains a basic flaw in how Janus treats direct-fire hit assessments. The flaw amounts to the fact that Janus does not use the appropriate AMSO model standard (the direct-fire hit assessment methodology from "The Compendium of Close Combat Tactical Trainer Algorithms, Data, Data Structures, and Generic System Mappings" (AMSAA [1996])).

Direct-Fire Hit Assessments

There are two fundamentally different approaches to direct-fire hit assessments that are currently used by the US Army in high-resolution Monte-Carlo combat simulations (whether they be for training or analysis purposes)

(1) **miss-distance distribution method**,

(2) $\mathbf{P_{SSH}}$ **method**.

These two methods yield identical results for the first round, but can differ appreciably for multiple-round engagements of a target by a particular firer.

Flaw in Janus

For multiple rounds fired in an engagement, the P_{SSH} method amounts to (since sampling will be independent in any Monte-Carlo procedure)

The above expression (in general) does not yield results exactly equivalent to the miss-distance-distribution method, because of the presence of so-called **variable bias** in weapon-system performance. Research is needed to determine conditions and weapons-system types for which this difference can be appreciable. In any case, AMSAA has extensive data to support either method (e.g. see AMSAA [1996]). The second (simpler) method is frequently used in high-resolution simulations, when run time is an issue. The first method, of course, yields theoretically correct results.

Suggested Change in Janus

If possible (and practically feasible), it is suggested that the miss-distance distribution method (as described in "The Compendium of Close Combat Tactical Trainer Algorithms, Data, Data Structures, and Generic System Mappings" (AMSAA [1996, Chapter 4]) be implemented in Janus for direct-fire hit assessment.

Reference

US Army Material Systems Analysis Activity (AMSAA), "The Compendium of Close Combat Tactical Trainer Algorithms, Data, Data Structures, and Generic System Mappings," Special Publication 74, Aberdeen Proving Ground, MD, June 1996.

Appendix 12: Flaw in Janus Direct-Fire Assessments

This updates the author's "Flaw in Janus Direct-Fire Assessments" (see Appendix 8 above). The additional information given here is an updated reference to standard Army algorithms used in high-resolution attrition modeling (AMSAA [1996]). The AMSO's Standards Coordinating Committee for Attrition (AMSO [1997,]) has proposed them as standard algorithms in the development of high-resolution simulations and simulators for distributed environments. The compendium's focus is primarily on ground combat, attack helicopters, and ground-based air defense. The areas addressed include vulnerability modeling and the physical aspects of attrition for various categories of weapon systems: direct-fire weapon systems, indirect-fire weapon systems, ground-based air-defense systems, and minefields. The behavioral and cognitive aspects of attrition are also included (AMSO [1997, p. 43]).

References

US Army Material Systems Analysis Activity (AMSAA), "Compendium of High Resolution Attrition Algorithms," Special Publication 77, Aberdeen Proving Ground, MD, October 1996.

Army Model and Simulation Office (AMSO), "Army Model and Simulation Standards Report FY98," Washington, DC, October 1997 (Available on AMSO World-Wide Website, with Homepage http://www.amso.army.mil.)

Appendix 13: Independent Rounds or Correlated Rounds?

Introduction.

This report has been critical of the use of the so-called independent-rounds model implicitly used by LLNL in JCATS. This appendix will attempt to briefly give some insight into the technical basis for this criticism. AMSAA has developed an excellent technical solution to this problem: namely, Monte Carlo every round. This solution is not only technically sound, but also very simple. Unfortunately, its very simplicity masks the underlying technical issue.

Background.

For direct-fire attrition, JCATS assesses firing outcomes by Monte-Carloing outcomes overtime to simulate the engagement kill probability. Since this Monte-Carlo procedure, draws independent samples from a (uniform) pseudorandom-number generator, this sampling procedure is equivalent to using the following formula and doing a single draw from the random-number generator.

where $P_K(n)$ denotes the engagement kill probability based on firing the n rounds at the target, and P_{SSK} denotes a single-shot kill probability (assumed to be constant over time). When the single-shot kill probability is allowed to change over time (e.g. through changes in the range between firer and target), formula (1) takes the form (still assuming independence between rounds)

where the subscript "j" on P_{SSK} denotes a particular round that has been fired. Use of this subscript allows one to play variations in P_{SSK} over time. However, the U.S. Army's "Engineering Design Handbook: Army Weapon System Analysis, Part One " (DARCOM [1977, p. 20-5]) says[5]

> **We should emphasize immediately that Eqs. 20-5 and 20-6[6] do not apply in general for multiple rounds. In spite of their almost universal use, they can be subject to serious errors in many applications not involving the rather strict assumptions that on the average the gunner has zero aim error but commits a shot-to-shot air error described by σ_μ, as we will see. Walsh (Ref. 1) indicates that for relatively small hit probabilities per shot, formulas of the type of Eqs. 20-5 and 20-6, the latter being of the Poisson type, may still apply with suitable accuracy even for occasions involving dependent events. Hence, such uses of Eqs. 20-5 and 20-6 should be checked independently as the occasion may require.**

In some very real sense, the "Engineering Design Handbook" provides the theoretical background for the attrition algorithms in AMSAA [1996a], [1996b], [2000]. Moreover, such AMSAA/BRL work has not led to simple formulas that clearly explain to the neophyte why the model (1) is inappropriate under many (if not most) operational circumstances. In the next section, an example is given (salvo fire) that can be used to show how bad an approximation (1) can prove to be.

The results in Chapter 20 "Multiple Round Hit Probabilities, Target Coverage, and Target Damage" of DARCOM [1977] primarily apply to artillery fire, traditionally a major concern of modern armies. There is little tie-in given for direct-fire weapons, although it certainly exists. The

[5] This document was primarily written by Dr. Frank E. Grubbs, formerly Chief Operations Research Analyst of the U.S. Army Ballistic Research Laboratories (BRL), prepared for the Engineering Design Handbook Office (prime contractor to U.S. Army Materiel Development and Readiness Command) (DARCOM [1977, p. xx]). BRL was the predecessor organization of AMSAA.

[6] This second equation cited here is an approximation to (1) that was widely used before computers were as wide spread as they are today. The first is simply equation (1) above.

important point to note here, however, is that there are no simple models and formulas for correlated rounds given in DARCOM [1977] (see also Eckler and Burr [1972, Chapter 2]). This is the underlying reason why essentially Monte-Carlo procedures are the only practical way of simulating multiple-round engagements when there is appreciable round to round correlation (and there invariably is, at least AMSAA data tells one). Moreover, this is the theoretical justification of the direct-fire attrition algorithm for non-automatic-fire modes given in "The Compendium of CCTT Algorithms" (AMSAA [1996a, Section 4.3.1], [2000, Section 4.3.1]). The fact the non-independent rounds are being considered is evident from the use of a "variable bias."

Before leaving this section, some useful notation for future comparisons will be established. Let us accordingly denote the engagement-kill probability (i.e. cumulative kill probability) for n independent rounds as

The engagement-kill probability for these n independent rounds is given by

Salvo-Fire Model.

The term "salvo" is used to denote the situation in which all n rounds are directed at the same aim point. The rounds are assumed to be independent of each other and all have the same delivery error. For simplicity in illustrating our point, the one-dimensional case will be considered here. Then the engagement-kill probability is given by

where the conditional single-shot kill probability is given by

The above notation will be explained below.

The assumptions made for this salvo-fire model are as follows

(1) target located at $x = 0$,
(2) common aim point, denoted as x_a, for salvo of n rounds; X_a is a random variable with mean 0, standard deviation σ_a, and density function denoted as
(3) delivery error D about aim point has mean x_a and standard deviation σ_d; the $i^{\underline{th}}$ round impacts at
(4) lethality function denoted as l(x),
(5) cumulative damage negligible.

If one assumes a so-called Gaussian lethality function (e.g. see DARCOM [1977, Section 15.6] or Taylor [1983, p. 141]), then the lethality function in (5) is given by
and it follows that the conditional single-shot kill probability is given by

Let us further assume that all distributions are normal (i.e. the distributions for aim error and delivery error). Substituting (6) into (4), using the binomial theorem, and carrying out the term by term integration, one obtains

It should be noted that for the above model the aim error, denoted as X_a, is realized only once for the salvo of n rounds, while the delivery error, denoted as D, is realized every round. Moreover, this aim error is in some sense equivalent to a target location error. This is an important point, since it allows one to interpret the above model as applying to the case in which the same realization of the target location error is used for all n rounds, whereas the case of independent rounds essentially means that the target is being aquired again independently after each round is fired. This latter point is key for understanding why AMSAA data does not support the independent-round model.

Summary of Results for the Two Models.

In this section, the results for the two models considered above are summarized. The **salvo-fire model** yields the following expression for engagement-kill probability

where the conditional single-shot kill probability is given by

For the case of a Gaussian lethality function and normal distributions for aim and delivery errors, one finds that the engagement-kill probability is given by (7).

The **independent-round model** yields the following expression for engagement-kill probability

where the single-shot kill probability is given by
and hence

where the standard deviation σ is the mean square error for aiming and delivery of fire, namely

Results of Numerical Computations.

This author has had students in classes at the Naval Postgraduate School do numerical experiments on the computer to compare the above two models. When the aim error (equivalently, the target-location error) is small relative to the delivery error, both (8) and (10) yield very similar results. However, when there is a relatively large aim error, there can be large discrepancies between the two formulas (8) and (10), with the independent-round model invariably yielding more optimistic results. In fact, the independent-round model can yield results several times larger than the salvo-fire model, even for as few as five rounds. In these cases, moreover, as n becomes large, the salvo-fire model does not even approach 1.0 asymptotically, but approaches a number less than one.

Discussion.

The above should provide some insight why the independent-round model (3) is simply a bad model for computing engagement-kill probability for many (if not most) cases of practical interest. Since most of the time targets are just not independently re-acquired after the firing of each round, one should not expect equation (10) to be a good model in all cases. Simple formulas were obtained above because of the assumption of Gaussian lethality, otherwise there are no such simple formulas in terms of conveniently tabulated functions. This is the reason for the Monte-Carlo procedure given by AMSAA [1996a], [2000] for direct-fire attrition (and identified by the occurrence of a variable bias). Furthermore, AMSAA has data that shows that the independent-round model (10) is simply a bad model for many (if not most) cases of practical interest.

References:

A.R. Eckler and S.A. Burr, Mathematical Models or Target Coverage and Missile Allocation, Military Operations Research Society, Alexandria, VA, 1972.

J.G. Taylor, "High-Resolution Combat Models (Micro-Combat Analysis)," Unpublished Class Notes Dated October 1983, Naval Postgraduate School, Monterey, CA, 1983.

U.S. Army Materiel Development and Readiness Command (DARCOM), "Engineering Design Handbook: Army Weapon System Analysis, Part One," DARCOM-P-706-101, Alexandria, VA, November 1977.

U.S. Army Materiel Systems Analysis Activity (AMSAA), "The Compendium of Close Combat Tactical Trainer Algorithms, Data, Data Structures, and Generic System Mappings," Special Publication No. 74, Aberdeen Proving Ground, MD, June 1996. (a)

U.S. Army Materiel Systems Analysis Activity (AMSAA), "Compendium of High Resolution Attrition Algorithms," Special Publication No. 77, Aberdeen Proving Ground, MD, October 1996. (b)

U.S. Army Materiel Systems Analysis Activity (AMSAA), "The Compendium of Close Combat Tactical Trainer Algorithms, Data, Data Structures, and Generic System Mappings," Special Publication No. SP-97, Aberdeen Proving Ground, MD, May 2000.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library, Code 013 Naval Postgraduate School Monterey, CA 93943-5100	2
3.	Research Office, Code 09 Naval Postgraduate School Monterey, CA 93943-5000	1
4.	James G. Taylor, Code OR/Tw Naval Postgraduate School Department of Operations Research Monterey, CA 93943	3
5.	Beny Neta, Code MA/Nd Naval Postgraduate School Department of Mathematics Monterey, CA 93943	3